Search Strategies for an Anytime Usage of the Branch and Prune Algorithm

Raphaël Chenouard, Alexandre Goldsztejn, Christophe Jermann

Search Strategies for an Anytime Usage of the Branch and Prune Algorithm

Raphael Chenouard
LINA, University of Nantes
France
Raphael.Chenouard@univ-nantes.fr

Alexandre Goldsztejn
CNRS, LINA
France
Alexandre.Goldsztejn@univ-nantes.fr

Christophe Jermann
LINA, University of Nantes
France
Christophe.Jermann@univ-nantes.fr

Abstract

When applied to numerical CSPs, the branch and prune algorithm (BPA) computes a sharp covering of the solution set. The BPA is therefore impractical when the solution set is large, typically when it has a dimension larger than four or five which is often met in underconstrained problems. The purpose of this paper is to present a new search tree exploration strategy for BPA that hybridizes depth-first and breadth-first searches. This search strategy allows the BPA discovering potential solutions in different areas of the search space in early stages of the exploration, hence allowing an anytime usage of the BPA. The merits of the proposed search strategy are experimentally evaluated.

1 Introduction

We consider numerical CSPs (i.e. variables domains are continuous, usually intervals of reals) with equality and inequality constraints. NCSPs with less equations than variables typically have continuous (infinite) solution sets. For example, the solution set of the NCSP

$$\langle\ (x,y)\ ,\ ([-2,2],[-2,2])\ ,\ \{x^2+y^2=1\}\ \rangle \qquad (1)$$

is the circle of radius 1 centered on $(0,0)$, a continuous set of dimension 1. This kind of NCSPs are met in numerous applications, e.g. design [Chenouard *et al.*, 2007] and robotics [Merlet, 2000].

Branch and prune algorithms (BPAs) [Van Hentenryck *et al.*, 1997] solve NCSPs alternating filtering and branching in order to explore exhaustively the search space following a search tree. Usually the stopping criterion is the size of the domains: they are processed until they reach a minimum size ϵ. The BPA hence returns a ϵ-paving, i.e. a sharp enclosure of the solution set made of ϵ-boxes (box domains whose size is smaller than ϵ). See different pavings of the CSP (1) on Figure 1. However, as soon as the solution set is too large (e.g. when the number of variables exceeds the number of equations by more than four), computing a ϵ-paving is impractical due to the huge number of boxes needed.

The BPA is however intrinsically anytime, since when stopped prematurely one still gets an enclosure of the solution set. But this premature paving is not very useful if the search-tree is explored depth-first (DFS) or breadth-first (BFS): DFS quickly converges to ϵ-boxes that are too close to one another to be representative of the solution set (see the left part of Figure 1); BFS computes a homogeneous paving but finds no ϵ-box at all if stopped too early (see the center graphic of Figure 1; note that such a sharp paving cannot be computed for larger solution sets, making BFS useless in such cases).

The search strategy used in an anytime BPA should quickly find ϵ-boxes that are *representative* of the solution set: ϵ-boxes should be discovered uniformly on a continuous connected component in the solution set, while every connected components should be reached by some ϵ-boxes in early stages of the search. Two such strategies are introduced in the present paper. The *most distant-first strategy* (MDFS) is introduced in Section 3.1; it consists in exploring the leaf of the search tree that maximizes the distance to the ϵ-boxes found so far. This strategy has good asymptotic properties (cf. Proposition 1), but lacks efficiency for quickly finding ϵ-boxes. The *depth and most distant-first strategy* (DMDFS) is introduced in Section 3.2; it is a greedy approximation of the MDFS, the latter being hybridized with a depth-first search to force a quick discovery of ϵ-boxes. Although the DMDFS does not possess the good theoretical properties of the MDFS, it shows a very good behavior on the presented experiments (cf. Section 4). The right part of Figure 1 shows the paving obtained using the DMDFS strategy. Six ϵ-boxes have been found instead of 56 using the DFS, but they are more *representative* than those obtained by the DFS.

Related Work

Recently [Goldsztejn and Granvilliers, 2008], an improved BPA has been proposed to tackle NCSPs with manifolds of solutions. However, this technique is based on the BFS and thus typically cannot be used to approximate solution sets of dimension higher than four or five.

When the solution set has a nonempty interior (typically when the NCSP involves only inequality constraints), BPAs can be improved by computing interior boxes, i.e. boxes that contain only solutions. However, the BPA still accumulates ϵ-boxes on the boundary of the solution set, a subspace having a dimension equal to that of the problem minus one. Thus, most of high dimensional solution sets remain out of reach of usual BPAs with interior box computation.

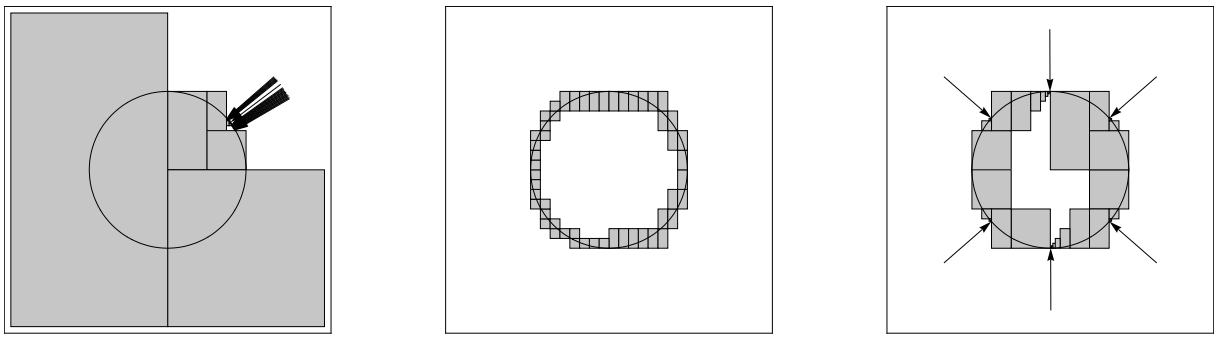

Figure 1: From left to right: The depth-first, the breadth-first and the depth and most distant-first strategies stopped after 100 bisections for the CSP (1). Each ϵ-box, for $\epsilon = 10^{-2}$, is pointed by an arrow (56 for DFS, 0 for BFS and 6 for DMDFS).

Interleaved DFS [Meseguer, 1997] consists in starting a DFS at each node obtained after a BFS limited to a reasonably low depth. However, this search strategy cannot give rise to an anytime algorithm: determining an adequate depth for the initial BFS is very difficult; too small it prevents local consistencies to be efficient enough, but too big it yields combinatorial explosion.

2 Interval Analysis

To tackle CSPs with continuous domains, a key issue is to prove properties on continuum of real numbers. Interval analysis handles this problem in an efficient way by using computations on floating point numbers. We recall here some basics which are required to understand the paper. More details can be found in [Neumaier, 1990; Jaulin *et al.*, 2001].

Interval analysis is mainly used to perform rigorous filtering through so-called interval contractors. An interval contractor for a constraint c on n variables with solution set $\rho_c \subseteq \mathbb{R}^n$ is a function $\mathrm{Contract}_c : \mathbb{IR}^n \longrightarrow \mathbb{IR}^n$ that satisfies the following relations:

1. $\mathrm{Contract}_c([x]) \subseteq [x]$;
2. $\forall x \left(x \in [x] \wedge x \in \rho_c\right) \Longrightarrow x \in \mathrm{Contract}_c([x])$.

Here, intervals are denoted with bracketed symbols and $\mathbb{IR}$ is the set of closed intervals. This definition is extended to a set of constraints $\mathcal{C}$ identifying $\mathcal{C}$ to the conjunction of its constraints. Hence, $x \in \rho_{\mathcal{C}}$ is true if and only if every constraint in $\mathcal{C}$ is true for x. Such contractors for standard equality and inequality constraints can be implemented using several techniques [Neumaier, 1990; Collavizza *et al.*, 1999] among which the $2B$–consistency (also known as hull consistency) [Lhomme, 1993] and the box-consistency [Benhamou *et al.*, 1994] play key roles in the resolution of numerical CSP. These filtering techniques are typically used in a branch and prune algorithm (BPA, described in Algorithm 1). This algorithm alternates filtering and branching so as to compute a sharp enclosure of the solution set. The output of the algorithm is a set of ϵ-boxes whose union contains all the solutions of the CSP. These boxes can be post-processed in order to check if they actually contain some solution[1]. The search strategy in the generic BPA is parametrized by the procedures

[1]this post-process usually applies to under-constrained systems after fixing enough variables in order to obtain a well constrained system, cf. [Kearfott, 2009]

Algorithm 1: Generic Branch and Prune Algorithm.

Input: $\mathcal{C} = \{c_1, \ldots, c_m\}$, $[\mathbf{x}] \in \mathbb{IR}^n$, $\epsilon > 0$
Output: $\mathcal{E} = \{[\mathbf{x}_1], \ldots, [\mathbf{x}_p]\}$

```
L ← ⟨Contract_C([x])⟩;
E ← ∅;
while L ≠ ∅ do
    ([x], L) ← Extract(L);
    if wid([x]) < ε then
        E ← E ∪ {[x]};
    else
        T ← Split([x]);
        T ← Map(Contract_C, T)[2];
        L ← Insert(L, T\{∅});
    end
end
return E;
```

Extract and Insert which handle the list $\mathcal{L}$ of boxes to be processed (current leaves of the search-tree). A DFS is implemented managing the list as a stack (LIFO), while a BFS is obtained managing the list as a queue (FIFO).

3 Search-tree Exploration Strategies

The goal of the proposed exploration strategies is to find ϵ-boxes in an order that is useful in an anytime algorithm, i.e., ϵ-boxes must spread across the search space in early stages.

3.1 The Most Distant-First Strategy

In order to spread ϵ-boxes across the search space, the search for a new ϵ-box should aim at maximizing the distance to the ϵ-boxes found so far. Formally, being given m ϵ-boxes $\mathcal{E} = \{[\mathbf{x}_1], \ldots, [\mathbf{x}_m]\}$, we wish the next ϵ-box $[\mathbf{x}_{m+1}]$ to maximize the cost function defined by

$$\underset{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]}{\mathrm{MinDist}}(\mathbf{x}) := \min\{d(\mathbf{x}, [\mathbf{x}_1]), \ldots, d(\mathbf{x}, [\mathbf{x}_m])\}, \quad (2)$$

i.e. to maximize the distance to the closest ϵ-box found so far. The distance d is an arbitrary distance between elements

[2]$\mathrm{Map}(f, \{x_1, \ldots, x_m\}) = \{f(x_1), \ldots, f(x_m)\}$.

of the search space. It is extended to sets considering the maximum distance between elements of these sets:

$$d([\mathbf{x}],[\mathbf{y}]) := \max_{\substack{x\in[\mathbf{x}]\\ y\in[\mathbf{y}]}} d(x,y) \tag{3}$$

which, though not a distance anymore, still satisfies the triangular inequality.

The most distant-first strategy (MDFS) is defined by choosing among all leaves of the search tree the one that maximizes (2). This is implemented in Algorithm 1 as follows:

(a) The boxes $\mathcal{T}\backslash\{\emptyset\}$ are inserted into $\mathcal{L}$ at Line 10 according to $\mathrm{MinDist}_{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]}$ so that the list is maintained sorted decreasingly.

(b) Each time a new ϵ-box is found, the list $\mathcal{L}$ is sorted decreasingly according to the new criteria $\mathrm{MinDist}_{[\mathbf{x}_1],\ldots,[\mathbf{x}_m],[\mathbf{x}_{m+1}]}$ just after Line 6.

(c) The first element of $\mathcal{L}$ is extracted at Line 4.

At the beginning of the search, no ϵ-box is yet available and $\mathrm{MinDist}_{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]}$ is undefined. The strategy is thus started with a depth-first search to quickly find out the first ϵ-box.

When ϵ tends toward 0, the $(m+1)^{th}$ ϵ-box found using the MDFS asymptotically converges to the actual most distant solution

$$\max_{\mathbf{x}\in\rho_{\mathcal{C}}} \underset{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]}{\mathrm{MinDist}}(\mathbf{x}) \tag{4}$$

To prove this property, we need the following lemma.

Lemma 1. *Provided that d is extended to sets using* (3)*, $d([\mathbf{y}],[\mathbf{z}]) \leq \epsilon$ implies $|\mathrm{MinDist}([\mathbf{y}]) - \mathrm{MinDist}([\mathbf{z}])| \leq \epsilon$.*

Proof. There exist some $i,j \in \{1,\ldots,m\}$ such that

$$\underset{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]}{\mathrm{MinDist}}([\mathbf{y}]) = d([\mathbf{x}_i],[\mathbf{y}]) \tag{5}$$

$$\underset{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]}{\mathrm{MinDist}}([\mathbf{z}]) = d([\mathbf{x}_j],[\mathbf{z}]). \tag{6}$$

Suppose, without loss of generality, that $d([\mathbf{x}_i],[\mathbf{y}]) \leq d([\mathbf{x}_j],[\mathbf{z}])$. We also suppose that $i \neq j$, the other case being similar and simpler. By the triangular inequality, we have $d([\mathbf{x}_i],[\mathbf{z}]) \leq d([\mathbf{x}_i],[\mathbf{y}]) + d([\mathbf{y}],[\mathbf{z}])$, which is less than ϵ by hypothesis, while by (6) we have $d([\mathbf{x}_j],[\mathbf{z}]) \leq d([\mathbf{x}_i],[\mathbf{z}])$. We have therefore proved that

$$d([\mathbf{x}_i],[\mathbf{y}]) \leq d([\mathbf{x}_j],[\mathbf{z}]) \leq d([\mathbf{x}_i],[\mathbf{y}]) + \epsilon, \tag{7}$$

which finally implies $|d([\mathbf{x}_i],[\mathbf{y}]) - d([\mathbf{x}_j],[\mathbf{z}])| \leq \epsilon$. □

Then, the following proposition shows that the MDFS asymptotically converges to arbitrarily sharp approximations of the global maximum of (4).

Proposition 1. *Consider Algorithm 1 where $\mathrm{Contract}_{\mathcal{C}}$ implements the global hull consistency. Let m ϵ-boxes $\mathcal{S} = \{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]\}$ be found by Algorithm 1 and suppose that at this point $\mathcal{L} \neq \emptyset$. Then the next ϵ-box $[\mathbf{x}_{m+1}]$ found using the MDFS contains a solution $\mathbf{x}$ which satisfies*

$$|d^* - \underset{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]}{\mathrm{MinDist}}(\mathbf{x})| \leq \epsilon, \tag{8}$$

where d^* is the global optimum of (4)*.*

Proof. The function $\mathrm{MinDist}$ will be used without the explicit reference to the ϵ-boxes $[\mathbf{x}_1],\ldots,[\mathbf{x}_m]$ in the sequel. First, we claim that there exist $\mathbf{x}^* \in [\mathbf{x}^*] \in \mathcal{L}$ such that $\mathrm{MinDist}(\mathbf{x}^*) = d^*$, and thus $\mathrm{MinDist}([\mathbf{x}^*]) \geq d^*$. This is due to the fact that every boxes of $\mathcal{L}$ are non-empty and hull consistent, and thus contain at least one CSP solution. Thus d^* obviously corresponds to a CSP solution that belongs to a box of $\mathcal{L}$ (the solutions belonging to $[\mathbf{x}_k]$ for $k \leq m$ having a lower $\mathrm{MinDist}$).

Now, because of the MDFS, when the ϵ-box $[\mathbf{x}_{m+1}]$ is extracted at Line 4, it satisfies

$$\forall [\mathbf{x}] \in \mathcal{L}\ ,\ \mathrm{MinDist}([\mathbf{x}]) \leq \mathrm{MinDist}([\mathbf{x}_{m+1}]). \tag{9}$$

Thus in particular $\mathrm{MinDist}([\mathbf{x}^*]) \leq \mathrm{MinDist}([\mathbf{x}_{m+1}])$ and $d^* \leq \mathrm{MinDist}([\mathbf{x}_{m+1}])$ is proved to hold. As $[\mathbf{x}_{m+1}]$ contains at least a solution $\mathbf{x}$, we have proved

$$\mathrm{MinDist}(\mathbf{x}) \leq d^* \leq \mathrm{MinDist}([\mathbf{x}_{m+1}]). \tag{10}$$

But finally as $\mathbf{x} \in [\mathbf{x}_{m+1}]$ and $\mathrm{wid}([\mathbf{x}_{m+1}]) \leq \epsilon$, we have $d(\mathbf{x},[\mathbf{x}_{m+1}]) \leq \epsilon$, and thus Lemma 1 proves that $|\mathrm{MinDist}(\mathbf{x}) - \mathrm{MinDist}([\mathbf{x}_{m+1}])| \leq \epsilon$ which together with (10) conclude the proof. □

The global hull consistency is necessary in Proposition 1. Though usual consistencies are not as strong as the optimal hull consistency, their efficiency increases as the width of interval decreases, and thus Proposition 1 is an asymptotic property of the MDFS for small ϵ.

Proposition 1 shows that the MDFS actually solves a constrained optimization problem to find the solution of the NCSP that maximizes the distance to the solutions found so far. Thus one can think of using other methods to solve this optimization problem. However, our experiments have shown that local optimizers, including genetic algorithms, do not converge to good enough solutions to be used in place of MDFS, while global optimizers are too slow.

3.2 Mixing the Most Distant and Depth-First Strategies

The BPA with a MDFS can be used as an anytime algorithm since it produces ϵ-boxes that are well distributed. However, as it solves a global optimization problem to find each new ϵ-box, it cannot be efficient in general. The search strategy proposed in this section is an approximation of the MDFS that finds representative ϵ-boxes much quicker. To this end, the MDFS is hybridized with the depth-first strategy, so as to keep the advantages of both approaches, i.e. a quick discovery of ϵ-boxes which are still representative of the solution set. The depth and most distant-first strategy (DMDFS) is defined by keeping the points (b) and (c) of the MDFS and by modifying (a) to

(a') The boxes $\mathcal{T}\backslash\{\emptyset\}$ are inserted at the beginning of $\mathcal{L}$ at Line 10 (LIFO).

Thus, the DMDFS search-tree is reorganized according to the distance to the ϵ-boxes found so far each time a new ϵ-box is found. An additional heuristic is used for choosing among the boxes at the same depth: the one that maximizes $\mathrm{MinDist}_{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]}$ is explored first. This is performed in (a')

sorting decreasingly w.r.t. $\mathrm{MinDist}_{[\mathbf{x}_1],\ldots,[\mathbf{x}_m]}$ the boxes in $\mathcal{T}\backslash\{\emptyset\}$ before inserting them in front of $\mathcal{L}$.

The DMDFS does not have as good asymptotic properties as MDFS: it implements a greedy optimization of (4) instead of a global optimization. The following example shows a situation where it does not converge to the global optimum.

Example 1. *Consider the situation depicted on the left of Figure 2: we have already found one ϵ-box (black box) and the problems has six remaining solutions $\mathbf{x}_1,\ldots,\mathbf{x}_6$ (black points) distributed into the two boxes to be processed (light gray boxes). The solution $\mathbf{x}_2$ is the one that maximizes the distance to the ϵ-box. However, the right hand side light gray box maximizes $\mathrm{MinDist}_{[\mathbf{x}_1]}$ (dashed line). Therefore, both the MDFS and the DMDFS process this box first, bisecting and filtering it, and finally give rise to the right hand side graphic of Figure 2 where three boxes remain to be processed.*

In this situation, the MDFS and the DMDFS operate differently: the MDFS processes the box that maximizes the distance to the ϵ-box among all three boxes, i.e. the light gray box, and will eventually converge to $\mathbf{x}_2$; while, the DMDFS processes the box that maximizes the distance to the ϵ-box among the boxes of largest depth only, i.e. dark gray boxes, and will eventually converge to $\mathbf{x}_6$.

However, experiments reported in Section 4 show that the DMDFS computes well distributed ϵ-boxes while being much quicker than MDFS.

3.3 Implementation Issues

Sorting operations performed in (a), (a') and (b) are very fast provided that the distance to the closest ϵ-box found so far

$$\min\{d([\mathbf{x}],[\mathbf{x}_1]),d([\mathbf{x}],[\mathbf{x}_2]),\ldots,d([\mathbf{x}],[\mathbf{x}_m])\} \qquad (11)$$

is stored together with boxes inside $\mathcal{L}$, and updated when necessary. The most expensive operation is the computation of the new distances to the closest ϵ-box in (b) : When a new ϵ-box $[\mathbf{x}_{m+1}]$ is found, the number

$$\min\{d([\mathbf{x}],[\mathbf{x}_1]),d([\mathbf{x}],[\mathbf{x}_2]),\ldots,d([\mathbf{x}],[\mathbf{x}_{m+1}])\} \qquad (12)$$

must be computed for each box $[\mathbf{x}]\in\mathcal{L}$. This must be done using the following identity:

$$(12) \;=\; \min\{\,(11)\,, d([\mathbf{x}],[\mathbf{x}_{m+1}]\}, \qquad (13)$$

which saves most of the computations reusing (11) for the computation of (12).

4 Experiments

This section presents experiments showing that the MDFS and the DMDFS both cover uniformly a simply connected solution set, and quickly reach all connected components of a solution set that has several ones. They will also illustrate Proposition 1 and demonstrate the practical efficiency of the DMDFS.

Experiments have been carried out on an 1.5 GHz Intel Pentium M based computer. The algorithms used for filtering are based on the interval library PROFIL/BIAS [Knueppel, 1994].

4.1 A Simply Connected Solution Set

We wish here to verify the repartition of the first solutions computed by the MDFS and the DMDFS inside one connected component. We consider the CSP $\langle(x,y),([-2,2],[-2,2]),\{11x^8-x^6-183x^4y^2+44x^6y^2+117x^2y^4+66x^4y^4-21y^6+44x^2y^6+11y^8\le 0\}\rangle$. This CSP has a flour shaped solution set with area π. The first 10^{-5}-boxes obtained with the MDFS and the DMDFS are shown on Figure 3. The MDFS behaves very well, while the DMDFS is less good at the beginning of the search though yielding quite homogeneously distributed 200 solutions.

More formally, the distance between the closest solutions among m solutions placed on a regular grid inside a square of surface π is $\sqrt{\pi/m}$. Therefore, a good repartition of m solutions (with m large enough) should have an average distance to closest neighbor that converges to this ideal distance. As shown by Figure 4, both the MDFS and the DMDFS reach this asymptotically good repartition, though the standard deviation is slightly better for the MDFS. On the other hand, computing the first 200 solutions took 27 seconds with the MDFS while only 5 seconds with the DMDFS.

4.2 Solution Set with Multiple Connected Components

We now wish to verify that the search strategies quickly reach all connected components of the solution set. To this end, we will measure the time needed to place at least one ϵ-box on each connected components of a scalable CSP whose solution set consists of n non-overlapping balls in a space of dimension n. The balls have radius 1 and a random center in $[-100,100]^n$. This solution set is naturally obtained with a disjunction of constraints, but also corresponds to the following conjunction of constraints:

$$\sum_{i=1}^{n}(x_i-c_{ji})^2-1 \;=\; y_j, \qquad (j=1..n) \qquad (14)$$

$$\prod_{j=1}^{n} y_j \;\le\; 0. \qquad (15)$$

We have varied n from 2 to 14 and measured the average time on 10 different random problems for $\epsilon=10^{-6}$. The MDFS already failed for $n=3$ and therefore does not appear in the following comparison. The DMDFS has been tested with both hull-consistency and box-consistency, although hull-consistency is foreseen to be more efficient on this NCSP since it has one occurence of each variable in each constraint.

A natural competitor to our strategies is the Monte Carlo approach, which here consists in randomly generating solution candidates in the variables domains. However, this random search must be hybridized with a local search since the probability of obtaining a solution at random is quasi-null. For the local search, two state of the art optimizers have been used, namely knitro[3] and donlp2[4]. In order to apply

[3] Available at `http://www.ziena.com/knitro.htm`.

[4] Available at `http://www-fp.mcs.anl.gov/OTC/Guide/SoftwareGuide/Blurbs/donlp2.html`.

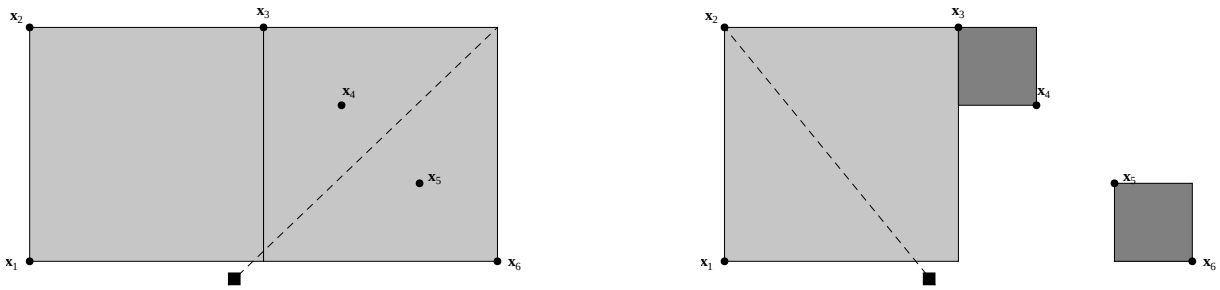

Figure 2: Situation where DMDFS does not converge to the same solution as MDFS.

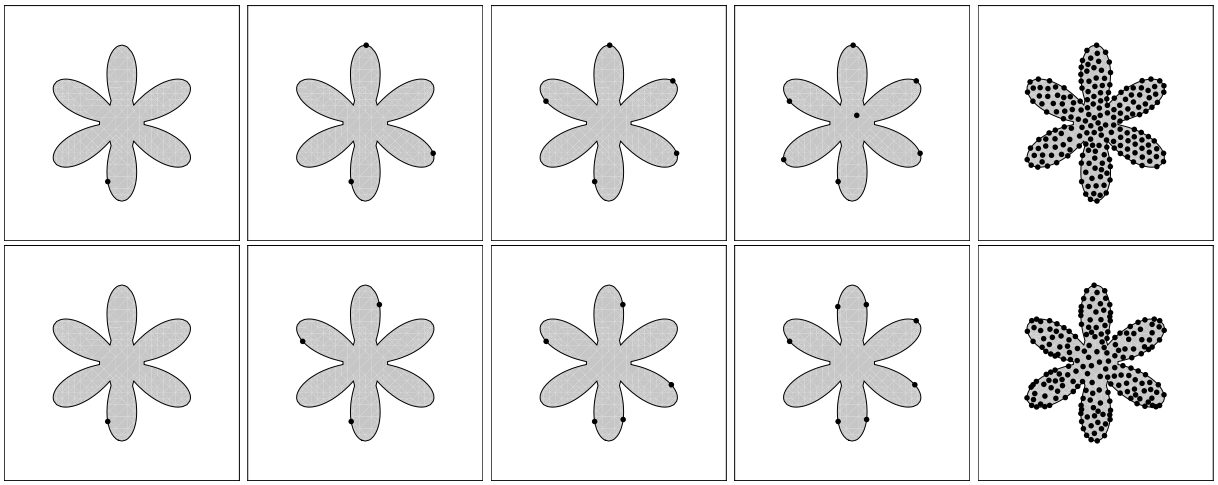

Figure 3: First 1, 3, 5, 7 and 200 solutions found using the MDFS (upper graphics) and DMDFS (lower graphics).

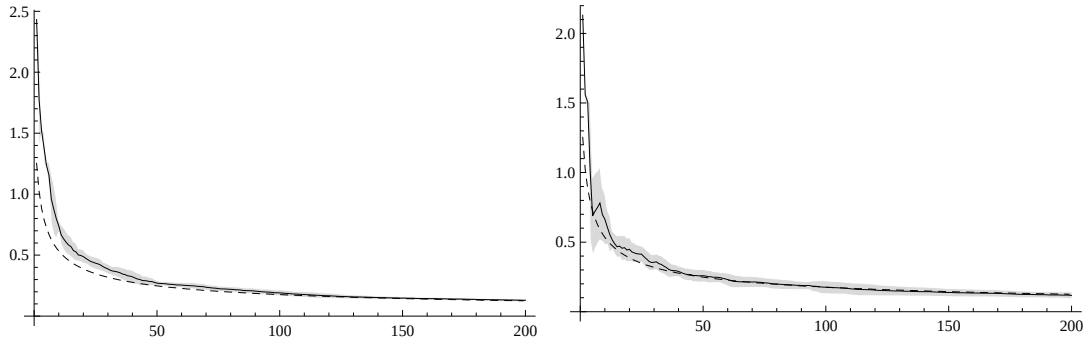

Figure 4: Average distance to the closest neighbor obtained enclosed inside the $\pm$ standard deviation hull, for the MDFS (left) and the DMDFS (right). The asymptotic ideal distance $\sqrt{\pi/m}$ is shown in dashed line.

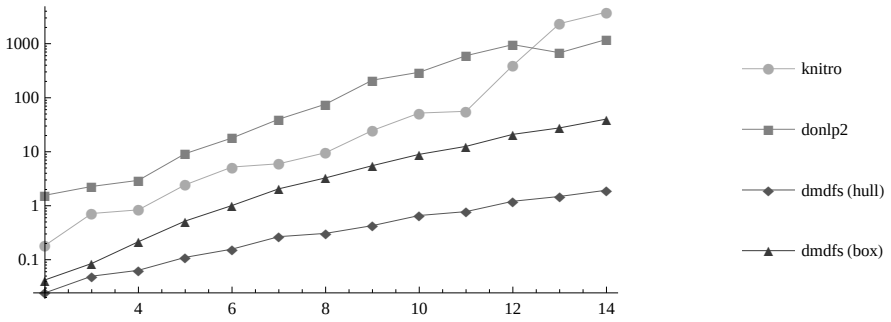

Figure 5: Time needed in seconds by each tested method to reach the n connected components of the Section 4.2 NCSP, w.r.t. n. As expected, hull-consistency provides better timings than box-consistency on these NCSP.

them to our problem, we have transformed our NCSP into an optimization problem either with a constant objective function when they accept equation constraints (donlp2) or with a least-square objective function on the violation of the equations when they do not handle equation constraints (knitro). Both methods have been used with their default settings after verifying that these settings prove to be optimal for the problem we address.

Figure 5 shows that the DMDFS allows tackling problems that are out of reach of random searches even when hybridized with efficient local searches.

5 Conclusion

Two new search-tree exploration strategies for branch and prune algorithms have been proposed. They both allow using the BPA as an anytime algorithm by spreading solutions across the solution set in early stages of the search. The most distant-first strategy (MDFS) has good asymptotic properties but is not very efficient in practice. The depth and most distant-first strategy (DMDFS) is a hybridization of the DFS and the MDFS that takes advantage of both. The DMDFS is very promising as it allows tackling NCSPs with large solution set while only a very few techniques are available for searching representative solutions of such CSPs. Reported experiments have confirmed the practical applicability of this search strategy and shown that local search approaches cannot compete on tested benchmarks. It is worthwhile noting that our strategies are not restricted to numerical CSPs since the BPA applies also in the discrete case. We thus plan to apply it to mixed discrete-continuous CSPs met in design problems and other applicative domains.

References

[Benhamou *et al.*, 1994] F. Benhamou, D. McAllister, and P. Van Hentenryck. CLP(Intervals) Revisited. In *International Symposium on Logic Programming*, pages 124–138, 1994.

[Chenouard *et al.*, 2007] R. Chenouard, P. Sébastian, and L. Granvilliers. Solving an Air Conditioning System Problem in an Embodiment Design Context Using Constraint Satisfaction Technique. In *Proceedings of CP 2007*, volume 4741/200 of *LNCS*, pages 18–32, 2007.

[Collavizza *et al.*, 1999] H. Collavizza, F. Delobel, and M. Rueher. Comparing Partial Consistencies. *Reliable Comp.*, 1:1–16, 1999.

[Goldsztejn and Granvilliers, 2008] A. Goldsztejn and L. Granvilliers. A New Framework for Sharp and Efficient Resolution of NCSP with Manifolds of Solutions. In *Proceedings of CP 2008*, volume 5202/2008 of *LNCS*, pages 190–204, 2008.

[Jaulin *et al.*, 2001] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, 2001.

[Kearfott, 2009] R.B. Kearfott. Interval Analysis: Verifying Feasibility. In C.A. Floudas and P.M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1730–1733. Springer, 2009.

[Knueppel, 1994] O. Knueppel. PROFIL/BIAS - A Fast Interval Library. *Computing*, 53(3-4):277–287, 1994.

[Lhomme, 1993] O. Lhomme. Consistency Techniques for Numeric CSPs. In *Proceedings of IJCAI 1993*, pages 232–238, 1993.

[Merlet, 2000] J.P. Merlet. *Parallel robots*. Kluwer, Dordrecht, 2000.

[Meseguer, 1997] P. Meseguer. Interleaved depth-first search. In *Proceedings of IJCAI 1997*, pages 1382–1387, 1997.

[Neumaier, 1990] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge Univ. Press, 1990.

[Van Hentenryck *et al.*, 1997] P. Van Hentenryck, D. McAllester, and D. Kapur. Solving polynomial systems using a branch and prune approach. *SIAM J. Numer. Anal.*, 34(2):797–827, 1997.